МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра МОЭВМ

ОТЧЁТ

по лабораторной работе №2

по дисциплине «Алгоритмы и структуры данных»

Тема: Иерархические списки

Студент гр.8304 _____ Воропаев А.О.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2019

Цель работы.

Изучить динамические структуры данных, иерархический список, бинарное коромысло. Попробовать реализовать специальный иерархический список.

Задание.

Вариант №5

Написать рекурсивную функцию или процедуру, формирующую линейный список номеров всех вхождений одного бинарного коромысла в другое.

Описание алгоритма.

Программа поддерживает два способа ввода (из консоли и из файла). При обычном запуске программы ввод производиться из консоли, если передать программе на вход путь до файла, то ввод будет производиться из этого файла. Пользователю нужно ввести сначала коромысло(k), в котором будет производиться поиск подкоромысла(p), затем нужно ввести непосредственно само подкоромысло. Был создан класс *krm* для описания структуры бинарного коромысла (описание класса см. в разделе «описание функций и структур данных»). Считывание производиться с помощью метода класса *k.krm::read_k(std::istream& input).* Данный метод посимвольно разбирает поток ввода. Если во время ввода встречается недопустимый символ, то выводится сообщение “Unexpected symbol in the input stream”. В ходе считывания заполняются структуры, содержащиеся в объекте класса *krm*, а именно структуры *Element* и *s_expr*(описание структур см. в след. разделе). Затем производится поиск подкормысла(p) в коромысле(k) с помощью метода *k.find_k(krm& k, std::string s).* Данный метод сравнивает длину и вес каждого плеча двух коромысел, если вместо значения веса коромысло сдержит еще

одно коромысло, то для него происходит рекурсивный запуск того же метода. Если найдено совпадение, то выводиться номер вхождения найденного коромысла в коромысло, в котором изначально производился поиск. Иначе выводится сообщение «No matches found».

Описание функций(методов) и структур данных программы:

Класс *krm* содержит следующие структуры:

```
typedef struct Element{
    int depth;
    struct krm* parent = nullptr;
    atom* left;
    atom* right;
    ~Element(){
        delete(left);
        delete(right);
    }
}elem;
```

Данная структура содержит поля для описания бинарного коромысла.

Depth – счетчик глубины коромысла(для «корня» коромысла depth = 0)

Parent – указатель на коромысло, находящееся выше данного(для корня коромысла значение равно nullptr)

Left/Right – структуры, содержащие описание левого и правого плеча бинарного коромысла(подробное описание структуры см. ниже)

~Element() – деструктор, зачищающий память.

```
struct atom {
    int len;
    std::variant<int,krm*>value;
    ~atom(){
        if(std::holds_alternative<krm*>(value))
            delete(std::get<krm*>(value));
    }
};
```

Структура, содержащая описание плеча бинарного коромысла.

Len – длина плеча

Value – переменная, типа std::variant, в которой содержится либо вес плеча, либо указатель на подкоромысло.

~atom() – деструктор, зачищающий память указателя, если он содержится в value

Конструктор класса:

```
krm(int n){
    e = new elem;
    e->depth = n;
    e->left = new atom;
    e->right = new atom;
}
```

Конструктор принимает значение глубины коромысла, выделяет память под объект класса, структуры описывающие его плечи.

Деструктор класса:

```
~krm(){
    delete(e);
}
```

Методы класса:

```
1.bool read_k (std::istream& input);
```

Метод предназначен для считывания бинарного коромысла. Данный метод посимвольно разбирает поток ввода. В ней также содержатся вызовы метода *make_atom*. Возвращает false при ошибке, true – иначе.

Input – поток ввода.

```
2.bool make_atom(std::istream& input, krm* y, int mode);
```

Данный метод вызывается из метода считывания и формирует плечо того коромысла, из которого был вызван. Если на этапе считывания значения веса плеча находиться символ ‘(’, то начинается формирование подкоромысла(выделяется память под новый объект класса, рекурсивно вызывается метод make_atom для левого и правого плеча). Возвращает false при ошибке, true – иначе.

Input – поток ввода, передающийся из метода считывания.

Y – указатель на коромысло, в котором требуется сформировать плечо

Mode – переменная указывающая, какое плечо формировать(1 – left|| 2 - right)

```
3.void print_k();
```

Метод, выводящий коромысло(в таком же виде, как указано в методических указаниях к лаб. работе)

```
4.void find_k(krm& p, std::string s, int& n);
```

Метод предназначен для поиска подкоромысла(p) в коромысле, от которого был вызван данный метод. Он также выводит номера вхождений при совпадении коромысел, если совпадений не найдено – выводится сообщение “#<номер коромысла>: No match was found in this one”.

P – ссылка на подкоромысло, поиск которого производиться в коромысле, от которого был вызван данный метод.

S – строка, которая накапливает номер вхождения для каждого последующего кормысла.

N – счётчик кол-ва найденных совпадений.

Выводы.

Был создан специальный иерархический список, в котором использовались некоторые принципы двусвязных и односвязных списков. Было получено понимание конструкторов, деструкторов и условного стиля написания программ на языке программирования C++.

Тестирование:

Если программе не было подано аргументов, будет запрошено ввести строку для обработки. Также есть возможность считывать данные из файла. Для этого требуется передать программе название файла с тестами в качестве параметра.

1.

```
Entered koromyslo:
((6 4)(4 ((13 ((13 356)(42 51)))(42 ((13 356)(42 51))))))
_____________________________
Entered podkoromyslo:
((13 356)(42 51))
_____________________________
Results of search:
121: Match!!!
122: Match!!!
```

2.

```
Entered koromyslo:
((6 ((13 ((13 356)(45 51)))(42 ((13 356)(42 51)))))(4 ((13 ((13 356)(42 51)))(48 ((13 356)(42 51))))))

_____________________________
Entered podkoromyslo:
((13 356)(42 51))

_____________________________
Results of search:
112: Match!!!
121: Match!!!
122: Match!!!
```

Входные данные	Выходные данные
((6 ((13 ((13 356)(45 51)))(42 ((13 356)(42 51)))))(4 ((13 ((13 356)(42 51)))(48 ((13 356)(42 51)))))) ((13 356)(42 51))	Results of search: 112: Match!!! 121: Match!!! 122: Match!!!
((((A)[B]))[[[B](A)]])	Result : True
((13 356)(42 51)) fewghb	Unexpected symbol in the input stream(read_k/first_symbol)
((6 ((13 ((13 356)(45 51)))(42 ((13 356)(42 51)))))(4 ((13 ((13 356)(42 51)))(48 ((13 356)(42 51)))))) ((3 5)(6 7))	Results of search: No matches were found
((1 6)(2 5)) ((1 6)(2 5))	Results of search: 1: Match!!!
1234	Entered string: 1234 contains invalid symbols
?><<DDD./ ;qe,g[q4g	Unexpected symbol in the input stream(read_k/first_symbol)
@(1 6)(2 ((1 6)(2 5)))) ((1 6)(2 5))	Unexpected symbol in the input stream(read_k/first_symbol)
((1 6)(2 ((1 6)(2 5)1)) ((1 6)(2 5))	Unexpected symbol in the input stream(make_atom/value)

((4 ((1 ((1 6)(2 5)))(2 ((1 6)(2 5)))))(6 9)) ((1 6)(2 5))	Results of search: 111: Match!!! 112: Match!!!

Исходный код

Main.cpp

```
#include <iostream>
#include <cstdlib>
#include <fstream>
#include "krm.h"

using namespace std;

int main(int argc, char* argv[]) {
    krm k(0);
    krm p(0);

    if(argc > 1){

        ifstream in(argv[1]);

        if(!in) {
            cout << "File can't be open" << endl;
            return 1;
        }

        ;
        if(!k.read_k(in))
            return 1;

        cout << "First entered koromyslo: ";
        k.print_k();
        cout << endl;


        if(!p.read_k(in))
            return 1;

        cout << "Second entered koromyslo: ";
        p.print_k();
        cout << endl;
    }

    else {

        cout << "Enter koromyslo" << endl;

        if(!k.read_k(cin))
            return 1;

        cout << "Enter podkoromyslo" << endl;
        if(!p.read_k(cin))
            return 1;
    }

    cout << "Entered koromyslo: " << endl;
```

```
    k.print_k(); cout << endl << "_____________________________" << endl;
    cout << "Entered podkoromyslo: " << endl;
    p.print_k(); cout << endl << "_____________________________" << endl;

    cout << "Results of search:" << endl;

    int n = 0;
    std::string s;
    k.find_k(p, s, n);

    if(!n)
        cout << "No matches were found" << endl;

    return 0;
}
```

Krm.h

```
#include <variant>
#include <string>


class krm {
public:

    typedef char base;

    struct atom;

    typedef struct Element{
        int depth;
        struct krm* parent = nullptr;
        atom* left;
        atom* right;
        ~Element(){
            delete(left);
            delete(right);
        }
    }elem;

    struct atom {
        int len;
        std::variant<int,krm*>value;
        ~atom(){
            if(std::holds_alternative<krm*>(value))
                delete(std::get<krm*>(value));
        }
    };        //end atom



    elem* e;

    krm(int n){
        e = new elem;
        e->depth = n;
        e->left = new atom;
        e->right = new atom;
    }

    ~krm(){
        delete(e);
```

```
    }

    bool make_atom(std::istream& input, krm* y, int mode);
    bool read_k (std::istream& input); // основная
    void print_k();
    void find_k(krm& p, std::string s, int& n);
};
```

Krm.cpp

```
#include "krm.h"
#include <iostream>

void krm::print_k(){

    std::cout << "((" << e->left->len << ' ';
    if (std::holds_alternative<int>(e->left->value))
        std::cout << std::get<int>(e->left->value);
    else {
        std::get<krm*>(e->left->value)->print_k();
    }
    std::cout << ")(";

    std::cout << e->right->len << ' ';
    if(std::holds_alternative<int>(e->right->value))
        std::cout << std::get<int>(e->right->value) << "))";
    else {
        std::get<krm*>(e->right->value)->print_k();
        std::cout << "))";;
    }
}

void krm::find_k(krm& p, std::string s, int& n) {


    std::string tmp(s);
    bool first,second;
    first = std::holds_alternative<krm*>(e->left->value);
    second = std::holds_alternative<krm*>(e->right->value);


    if(first) {//Проверка левого плеча на содержание подкоромысла
        s.push_back('1');
        std::get<krm*>(e->left->value)->find_k(p, s, n);
    }


    if(second) {//Проверка правого плеча на содержание подкоромысла
        tmp.push_back('2');
        std::get<krm *>(e->right->value)->find_k(p, tmp, n);
    }


    if(!second && !first && std::get<int>(e->left->value) ==
std::get<int>(p.e->left->value)
        && e->left->len == p.e->left->len
        && std::get<int>(e->right->value) == std::get<int>(p.e->right->value)
        && e->right->len == p.e->right->len){
        std::cout << '1' << s <<  ": Match!!!" << std::endl;
        n++;
    }
}
```

```
bool krm::make_atom(std::istream& input, krm* y, int mode) {


    char v;
    std::string str;
    char* ptr;

    while (true) {
        input >> std::noskipws >> v;
        if (v >= '0' && v <= '9') {
            str.push_back(v);
            continue;
        }
        if (v == ' ')
            break;
        else {
            std::cout << "Unexpected symbol in the input
stream(make_atom/len)" << std::endl;
            return false;
        }
    }

    if(mode == 1)
        y->e->left->len = std::strtol(str.data(), &ptr, 10);
    else if(mode == 2)
        y->e->right->len = std::strtol(str.data(), &ptr, 10);

    str.clear();

    while (true) {
        input >> std::noskipws >> v;
        if (v >= '0' && v <= '9') {
            str.push_back(v);
            continue;
        }
        else if (v == '(') {
            krm* lower = new krm(y->e->depth + 1);
            input >> v;
            if(mode == 1)
               y->e->left->value.emplace<krm*>(lower);
            else if(mode == 2)
                y->e->right->value.emplace<krm*>(lower);

            lower->e->parent = y;

            if(make_atom(input, lower, 1))
                do input >> v; while(v == ' ');
            else
                return false;

            if(!make_atom(input, lower, 2))
                return false;

        } else if (v == ')') {
                if(mode == 2)
                    input >> v;
            break;
        }
        else{
```

```
            return false;
        }

    }

    if(v != ')'){
        std::cout << "Unexpected symbol in the input stream(make_atom/value)"
<< std::endl;
        return false;
    }



    if(mode == 1){
        if (std::holds_alternative<int>(y->e->left->value))
            y->e->left->value.emplace<int>(std::strtol(str.data(), &ptr,
10));
    }
    if (mode == 2) {
        if (std::holds_alternative<int>(y->e->right->value))
            y->e->right->value.emplace<int>(std::strtol(str.data(), &ptr,
10));
        }

    return true;
}


bool krm::read_k(std::istream& input){

    char x;
    do input >> x; while (x == ' ');
    if(x != '(')
        std::cout << "Unexpected symbol in the input
stream(read_k/first_symbol)" << std::endl;
    input >> x;

    if (x == '(' && make_atom(input, this, 1)) {
        do input >> x;
        while (x == ' ');
    }
    else{
        return false;
    }



    if (x == '(' && make_atom(input, this, 2)) {
        input >> x;
    }
    else{
        return false;
    }

    return true;
}
```